Why Teach Unix?

Bernard Doyle & Raymond Lister

Faculty of Information Technology
University of Technology, Sydney
Jones St. Broadway NSW 2007

bjd@it.uts.edu.au raymond@it.uts.edu.au

Abstract

This paper examines computing academics' conceptions of the Unix operating system, and the purpose of teaching Unix. Interview transcripts from nine academics were analysed phenomenographically. A small number of qualitatively different conceptions of Unix were identified, within two broad categories. The first broad category manifested a technical approach to Unix. Within this broad category, the conceptions of Unix were, from the least to most sophisticated − (1) Unix as a set of unrelated commands; (2) Unix as a command line interface superior to GUIs; and (3) Unix as a problem solving tool. The second broad category was a non technical conception of Unix, in which Unix was seen as a resource that is cheap, secure and robust. With regard to teaching Unix, two broad categories of reasons were identified − practical and pedagogical. These results for teachers are broadly consistent with an earlier phenomenographic study of student conceptions of Unix.

Keywords: Phenomenography, Unix.

Introduction

There have been major changes in the content of IT courses over the last 15 years. The ubiquity of personal computers running Microsoft windows, the impact of the internet as well as the widespread adoption of object oriented programming languages such as Java and C++ have meant that the content of computing degrees has altered radically. There has been a trend away from subjects dealing with low level technical details in favour of those with a high level approach or a managerial perspective. Courses in assembler languages, logic and discrete mathematics, compiler construction and computer hardware are now taught as electives or have become the responsibility of engineering faculties. The teaching of operating systems in general, rather than teaching a specific operating system, is now usually an elective, if it is offered at all. The change in the content of contemporary computing studies at university level is also reflected in the fact that universities now offer students choices in degrees of Information Technology, Computer Science, Software Engineering and Information Systems.

With the downturn in student enrolments, many Australasian universities are redesigning their degrees, in the hope of attracting more students. For academics participating in such redesigns, the stakes are high. Topics that some computing academics have loved and taught for many years are being removed as part of degree redesign, to make room for new topics.

Not all the change is one way. There is a "back to basics" movement, which advocates reversing some of the recent changes in computer education replace. For example, there has recently been a vigorous debate on the teaching of the first programming subject, with one side advocating a change back from teaching objects-early to the traditional procedural approach (Astrachan et al., 2005; Bruce, 2005; Reges, 2006). At least one Australian university has done exactly that, changing from C to Java as the first language taught, but subsequently changing back to C.

Lewis and Smith (2005) have placed these sorts of debates into a broader framework, arguing that members of the computing education community tend to debate curriculum issues from within three main conceptual frameworks – segregationist, integrationist, and synergist. Academics within the first of those frameworks argue for a traditional computer science syllabus, academics in the second frameworks argue for change while those in the third framework argue that syllabi should incorporate both traditional topics and new topics.

In the case of Unix, the debate is not so much about whether Unix should be taught at all − it appears most academics believe it should be taught − but instead the debate is about the depth to which Unix should be taught in redesigned degrees, and also the style of instruction that should be used to teach Unix.

In this paper, the authors do not argue their own position in the Unix debate. Instead, we seek to document and formalise the various views on Unix and its teaching. In particular, we seek to make explicit what often remains implicit in the heat of committee room debate. Our intention is similar to that of McCauley (2004) who, within the context of a different syllabus debate, advocated that participants need explicit agreement on terminology, so they can "*clearly and succinctly express what they had tried to say, previously, using plain English*". We believe that most syllabus debates would benefit from scholarly analysis of the debate itself.

1.1 Prior study of student conceptions of Unix

In an earlier paper, the authors conducted a phenomenographic study of conceptions of Unix among students attending the authors' university (Doyle and Lister, 2006). In that study, we noted that students compartmentalized their appreciation of Unix. For example:

"… students appear to see the superior security of Unix as an "accidental" property of Unix, not a consequence of the architecture of Unix. Perhaps, as we collect more interview transcripts, we will see students who do articulate such a connection. On the other hand, perhaps such a connection is not currently being articulated by the teachers".

In this paper, we present a new phenomenographic study, which is similar to that prior study, but in this study we examine academic teachers' conceptions of Unix. We also study the teachers' understandings of the purpose of teaching Unix in contemporary computing degrees. This study addresses the above speculation from the earlier study, that perhaps students are not making certain connections in their conception of Unix because those connections are not commonly being articulated by their teachers

1.2 Phenomenography

Phenomenography is a qualitative research technique which looks at the different ways people perceive, conceptualise, approach or understand a phenomenon. (Ackerlind, 2005). Usually, phenomenographic data consists of interview transcripts. The data is analysed to identify the qualitatively different ways in which the phenomenon is conceived. These different ways of knowing are referred to as the Categories of Description. The interrelationships between the categories define what is known as the outcome space. This space is often linear. That is, there is a single aspect that varies qualitatively across the categories. In such a linear space, the categories often form a hierarchy, with higher categories being more sophisticated conceptions that subsume the lower conceptions.

Phenomenography is widely used as an education research technique. It has been used to analyse student's conceptions of various academic disciplines such as Music (Reid, 1997), Physics (Booth and Ingerman, 2002) and Statistics (Reid and Petocz, 2003). It has also been used to analyse academic's approaches to teaching (Bruce & Gerber, 1995, Trigwell & Prosser, 1997, Trigwell, 2000).

Within computing, phenomenography has been used to analyse student's conceptions of TCP/IP (Berglund, 2005), Object Oriented Information System Development (Box and Lister, 2005), Learning to Program (Booth, 1992, Booth, 2001, Bruce *et al*., 2004, Stoodley *et al*., 2004, Eckerdal & Thun, 2005), and Information Systems (Cope, 2003). Phenomenography has also been used to analyse approaches to teaching computing topics in general (Lister et al., 2007) and the teaching of Data Structures (Lister et al., 2004).

2 Method

2.1 Interviewee Background

We interviewed nine academics in the Faculty of IT at the authors' university. The faculty consists of three departments. These are Information Systems, Software Engineering and Computer Systems. In order to obtain as broad a sample of views as possible, our interviewees were drawn from all three departments. Two came from Information Systems, five from Software Engineering and two from Computer Systems. A number of other academics were approached, but declined to be interviewed. Of the nine academics interviewed, three made moderate to extensive use of Unix in the courses they taught. Of the remaining six interviewees, four used it occasionally and the remaining two never used Unix at all in their teaching. All academics interviewed had some Unix experience either as undergraduates, postgraduates, in industry or in teaching. In some cases the exposure had occurred some time ago. For example, one interviewee had used the "vi" editor and some Unix commands to teach Cobol Programming over 20 years ago.

2.2 Interview Structure

Following standard phenomenographic procedures, the interviews were semi-structured and used the following question set. This had been prepared prior to the interviews.

1. Tell me about your experience with Unix.
2. What does the word "Unix" mean you to you?
3. In what ways are Unix and Microsoft Windows different?
4. In what ways are Unix and Microsoft Windows the same?
5. Is there any task for which you'd prefer to use Unix over Microsoft Windows?
6. Does Unix have any role in the subject you teach? If so, what is that role?

7 (a). What do you think is the role of Unix in an undergraduate IT degree?

7 (b). What do you think is the role of Unix in an undergraduate Computer Science degree?

7 (c). What do you think is the role of Unix in a postgraduate IT degree?

8. What do you think is the role of Unix in computing in general?
9. What do you think is the the role of an operating system, whether it be Unix, Windows, or any other operating system?
10: What do you understand by the Unix term "process"?
11: What do you understand by the term "file system"?
12: What type of tasks would you prefer to use command line for instead of a GUI?
13: What do you understand by the term "script"?
14. In what ways are scripting languages and application level programming languages different?
15. In what ways are scripting languages and application level programming languages the same?

16. What do you understand by the term "pipe"?

As part of the semi-structured interview process, the interviewer often asked follow-up questions immediately after individual prepared questions. This was done to illuminate interesting issues arising from the answers to the prepared questions. Approximately 70% of the questions were follow-up questions.

2.3 Analysis Technique

The data was analysed using standard phenomenographic techniques. In terms of the categorisation of phenomenographic approaches by Ackerlind (2005) our analysis used the following approach:

(1) We considered excerpts from transcripts.

(2) The first author analysed the data initially.

(3) The two authors then analysed the data jointly. This analysis focussed on attempting to resolve the initial independent interpretation of the first author and possible other interpretations of the data.

(4) The structure of the analysis was driven by the data, but obviously the authors were influenced by their previous phenomenographic study of student conceptions of Unix.

(5) The focus was on pragmatic validity. That is, the aim of the analysis was to provide insight into the teaching and learning of Unix

3 Results Part 1: Conceptions of Unix

3.1 Overview

Among the transcripts of the nine academics interviewed, we identified the following conceptions of Unix:

1. Unix as a command line interface.

This first conception consisted of two sub-categories:

   (a) An unrelated set of commands that is hard to learn and use.

   (b) A more powerful alternative to the Windows graphical user interface (GUI).

Two other conceptions of Unix identified were:

2. Unix as a tool for solving certain problems
3. Unix as a resource

These conceptions are discussed in greater detail in the following subsections.

3.2 Unix as a Command Line Interface

As summarised above, interviewees who manifested this conception of Unix tended to display one of two positions, discussed below.

3.2.1 An unrelated set of commands that is hard to learn and use

The following excerpts from teachers' transcripts illustrate this first position:

"*....but a command line interface which is the thing that really I think of first when I think of Unix, is just intolerable, it's something completely artificial and arbitrary. It's not even as useful as learning ancient Greek....* " (A5)

"*...Yes, well it also means a real pain in the neck operating system. It uses a command language. It's really annoying to use. It's really obscure. ... The kind of way that it only gives you feedback if anything goes wrong, so you are never quite sure that anything happened*." (A9)

3.2.2 A more powerful alternative to the Windows graphical user interface (GUI)

The second position views a command line interface as being closer to the underlying machine, and hence the command line interface has powers unavailable to a GUI:

"[The machine is] ... *accessible in the sense that you when you use it, you can get to it, you can drill down to the lowest level* [of the machine] *if you want to*" (A2)

"[Unix is] *an operating system that you interact with at a command line level rather than a GUI. You interact with* [the computer] *more directly than using something like windows which has a GUI on top of it*." (A3)

The above two interviewees were aware of the existence of Graphical User Interfaces for Unix, such as X windows, but these interviewees had a conception of Unix as a very effective operating system because it could be used at the command line level.

3.3 Unix as a tool for solving certain problems

Several academics saw Unix as possessing attributes that made it a useful tool for solving problems. These academics sometimes illustrated the power of Unix by describing why they chose Unix to solve problems within their own teaching. For example:

"*The software that I have written for taking student submissions and stuff has probably been a lot easier to write, its command line driven, but it's probably been a lot easier to write than it would be if I had written it under some windows environment. ... from a systems administration point of view very well,* [Windows is] *very awkward, whereas Unix is a lot more straightforward. There are still things I've had to find out with Unix which has frustrated me at times because I've had to figure my way around them, but I suspect I would have had a lot more trouble if I had to try and do this stuff under windows."* (A1)

3.4 Unix as a resource

The conceptions of Unix that we have described up to this point have been technical in their orientation. A non-technical but common conception is Unix as a resource. This conception focuses on useful properties of Unix that can be appreciated without necessarily having a strong technical background in Unix − a management perspective. Such properties are: robustness, speed, security, server hosting capability, networkability and cost (the last at least for open source versions).

[Unix] *has a better reputation for security and performance*." (A4)

"[Unix] ... *for me it means reliability.*" (A6)

"*Well, if you think about Linux, and people want to save a bit of money, maybe it's got a role in organisations that don't want to spend a huge amount of money on their software....* (A5)

3.5 Conceptions of Unix: The Outcome Space

In the previous three subsections we have described four categories in which the interviewees conceived of Unix. We will now look at how these categories relate to each other, to form an Outcome Space.

Three of the categories are technical in their orientation and form a hierarchical relationship. Among these three hierarchical categories, the lowest is the conception of Unix as a weakly or totally unrelated set of commands. Above that conception is the conception of Unix as a powerful command line interface which is available as an alternative to GUIs. The higher of these two conceptions differs from the lower because the higher category introduces a more unified view of the commands, as the command line interface is seen as offering better access than a GUI to the underlying machine.

The third and highest conception in this hierarchy is the conception of Unix as a tool for solving certain problems. In this highest conception, the degree of relatedness between the Unix commands is so great, the conception is of a single, unified tool kit. In this highest category, the focus has shifted away from the command line interface itself, to the problems that can be solved with the command line interface.

This hierarchy of three technical conceptions can be interpreted in terms of the SOLO taxonomy (Biggs & Collis, 1982). The type of interviewee response that illustrates the lowest conception is a unistructural response. The type of interviewee response that illustrates the intermediate conception is a multistructural response, while the highest conception manifested in a relational response.

At this stage of the project, with the interview data currently available to us, the fourth and non-technical category − Unix as a resource − cannot be related to the hierarchy formed by the three technical categories.

4 Results Part 2: Why Teach Unix?

All nine interviewees thought that learning Unix should be compulsory, at least for Computer Science students. Seven of the nine thought Unix should also be a compulsory part of an IT degree. In our interview script, we did not explicitly pursue the issue of just how detailed a treatment of Unix should be taught in such degrees.

The reasons cited for teaching Unix fell into two broad classes.

- Practical
- Pedagogical

The difference between these two categories is that the practical category focuses upon the computing environment in which the student will eventually work, whereas the pedagogical category focuses upon the student, and the intellectual development of the student. These understandings are not mutually exclusive, and interviewees frequently manifested both understandings.

Each understanding is described in greater detail in the following two subsections.

4.1 Practical Reasons

In this category, the understanding of the purpose of teaching Unix is that it is an essential skill for computing professionals:

"*Certainly I think that as a graduate student, if the company took them out and stuck them in front of a Unix terminal, the students should be able to go 'OK, I'm not terrified of this...*' " (A1)

"*So I guess, at least if students have enough of a flavour of it so they don't get out there and say 'Oh! What's Unix?' when they went to a Unix shop, because that would make them look a bit silly. And I guess the other thing is...* [any organization] *has to have a main operating system that they use .... if it happens to be Unix in the place that they work in then they probably should know about it*." (A9)

The fact that Unix is used widely on the internet was given as a more specific reason why Unix should be taught. For example:

"*If they are going to understand the internet and a lot of the internet functionality works, they will need to understand how Unix works. To understand a lot of the hardware, everything from routers to switches to hubs that makes the internet operational, they will need an appreciation of the kind of programming that needs to be done, the kind of operating systems that are tailored, some of them based on Unix variations that are installed on those systems."* (A9)

4.2 Pedagogical Reasons

Unix, it is argued in this conception, expands the student's horizons. Unix is seen as an alternative model to Microsoft Windows, with which the students are more familiar. The interviewees were not necessarily hostile towards Microsoft Windows or GUIs in general, but they argued for breadth in student education:

"I think they also need to have diversity in operating systems ... [so that] *... they don't think the world just consists of windows. That there are other operating systems"* (A2)

"... I wouldn't see Unix as the primary vehicle for teaching, but for developing a deeper understanding of what they are doing at that level of Graphical User Interface, and then they see the result of that under the hood. It's important to create that understanding, - how computers work, what's happening in there …" (A7)

"I suppose there are two issues. One would be as a kind of example or historical kind of thing to say this has been a very influential type of operating system and these are what some of the features are and this is why it's so popular at the feature level, and these are where its' shortcomings are, that other people might want to add things in." (A9)

4.2.1 Knowing "what's under the hood"

One interviewee expressed the view that computer science graduates, but perhaps not IT graduates, should have an appreciation of the internals of Unix:

"*In a CS degree I think the role of Unix is still major, and it would be very important to talk about the architecture of Unix ... a famous book* [see Bach, 1986] *.... describes*

everything that happens inside Unix, the Unix kernel, how it works, why it uses certain data structures, how does it pass those data structures in an efficient manner, how does it schedule processes …. In a Computer Science degree I think this would be a major role. In Information Technology sort of degree you would have to consider that students might like to use it, not necessarily understand it." (A4)

Another interviewee took the importance of "knowing what's under the hood" even further:

"[Open source software] *gives students the opportunity to be exposed to a lot more software at the code level ... whereas they are unlikely to look at the code of proprietary* [software]." (A3)

We list this open source argument here, under pedagogical reasons, because this particular argument for open source does not rest upon the software being free, but on the educational opportunities that flow from having access to source code.

5 Discussion

5.1 Sample Size

Inevitably, a good portion of the readers will be troubled by the small sample size for this phenomenographic study (i.e. nine academics).

Phenomenography is a qualitative research method, not a quantitative method. From the data presented, it would not be appropriate to speculate upon the popularity among academics of any of the above categories. To make such conclusions would require significantly more data and a different research method. The aim of phenomenographic research is merely to capture the full spectrum of diversity, not quantify it.

While small sample sizes may be compatible with some qualitative methods, in the study presented in this paper we do not claim that nine interview transcripts is sufficient for final conclusions to be drawn. It is possible to perform a preliminary phenomenographic analysis with only nine interviewees, and we present our results as preliminary. We make no claim to have identified, at this stage, the full spectrum of diversity in academics' conceptions of Unix. However, while interviewing more subjects may elaborate upon our preliminary analysis, we are confident that further data will not invalidate this preliminary study. That is, interviewing more academics will probably add more categories, add further structure to the outcome space, and perhaps refine the category definitions, but collecting more interviews is unlikely to completely invalidate the categories and outcome space as we have identified it in this paper. Having collected a subset of the data we will eventually collect, we believe our existing analysis captures a subspace of the outcome space we will eventually construct.

One strong reason for having confidence in this preliminary analysis is that the results are compatible with our earlier phenomenographic analysis of student conceptions of Unix. We compare the results of these two studies in the next subsection.

5.2 Relationship to prior study of students

In both this study of teachers and the authors' previous study of students (Doyle and Lister, 2006), interviewees clearly articulated the same category "Unix as a resource". Also, both sets of interviewees articulated a linear hierarchical set of technical conceptions. Both of these hierarchies contained three conceptions of Unix. However, while the actual categories within the two hierarchies are similar, the categories are not identical. Table 1 summarises and compares the linear hierarchical set of technical conceptions from the two studies. The remainder of this section compares these conceptions in detail.

<table>
<tr><th>Sophistication</th><th>Teachers</th><th>Students</th></tr>
<tr><td rowspan="4">High

Low</td><td></td><td>A professional computing environment</td></tr>
<tr><td colspan="2">A tool for solving certain problems</td></tr>
<tr><td>A more powerful alternative to the Windows GUI</td><td></td></tr>
<tr><td colspan="2">An unrelated set of commands</td></tr>
</table>

Table 1: A comparison of the linear, hierarchical technical conceptions of Unix, for the teachers in this study and the students from the prior study

The bottom and least sophisticated conception of Unix is the same for both teachers and students − Unix as an unrelated set of commands.

In the teachers' conceptions of Unix, the intermediate category (a more powerful alternative to the Windows GUI) is not apparent in the transcripts of the student interviews. We suspect this is because this academic conception rests on the notion of an underlying machine, but the students (who are in their first year of study) know very little about the underlying machine.

Both teachers and students share the next level in the hierarchy (Unix as a tool for solving certain problems), but that is the highest category for the academics, whereas the students show another conception, Unix as a professional computing environment. We have, for this paper, tentatively placed it as a higher category, but at this time we do not understand this category well. Some students transcript excerpts that we placed in this category are:

"*I think Unix and Linux is more powerful. …. It's more professional than Microsoft … I discovered a new world of computing in Unix. ….*" (S01)

"*I am a systems administrator, and I used to use Microsoft based, and we have, you know, so many problems, with Microsoft, if you are a system administrator. Unix now, opens, I think a new track for me, to deal with system administration using Unix, and now I am planning to study Unix systems administration next semester, because I would like to be a Unix systems administrator. Because I like …*

[Unix] *... so much. I think it's powerful, and will develop my future career in systems administrator*." (S03)

It may be that these students are articulating what is really the same conception as the top academic category, but they express it this way because are focussed on their chosen future in industry.

While there may be some difference in the categories and outcome space identified in the two studies, the results are very similar. This is not surprising. The academics and students interviewed are all from the same university. Therefore, the student conceptions reflect those of their teachers.

5.2.1 Unix as a resource

As part of that prior study of students, the authors wrote the following:

"At this stage of the project, it appears that students do not connect the category "Unix as a resource" to the other three categories. For example, the students appear to see the superior security of Unix as an "accidental" property of Unix, not a consequence of the architecture of Unix. Perhaps, as we collect more interview transcripts, we will see students who do articulate such a connection. On the other hand, perhaps such a connection is not currently being articulated by the teachers".

In our interviews with academics, with only one exception, the academics did not articulate such a connection to us. It seems likely, therefore, that these teachers also do not articulate such a connection to their students.

5.3 Validity and Reliability

As with all phenomenographic studies, the categories that we have inferred from our data are probably not the only categories that can be inferred from the data. However, if we presented both our interview transcripts and our categories of description to other phenomenographers, they should agree that our categories of description can be inferred from our data. This is known as communicative validity (Ackerlind 2006)

The reason why alternate sets of categories are possible is that the categories identified in any phenomenographic study are to some extent dependent on the intent of the phenomenographer. Our intent is to facilitate debate on the teaching of Unix, and we chose our categories accordingly. In formal terms, our aim is to produce results which exhibit pragmatic validity (Ackerlind 2006). Our research aim is to provide insights that may be used in the design of courses on Unix, and we believe the results we have presented fulfil the criteria for pragmatic validity.

5.4 Relation between Conceptions of Unix and the purpose of teaching it

It is reasonable to expect that there is a relationship between how academics' conceive of Unix, and how they understand the reasons for teaching it. For example, it seems reasonable that an academic who tends to see Unix as an unrelated set of commands might also be drawn to pragmatic understandings of why it should be taught, and not drawn to believe that students need to understand "what's under the hood". This may be the case, and further research may confirm that hypothesis, but at this time there is insufficient data to confirm or refute such a conjecture.

6 Conclusion

In this study of why Unix is taught, two broad categories of reasons were identified from interviews with academics – practical and pedagogical. Given the small sample size (even by the standards of phenomenographic research) we present these finding as preliminary. In future work, we will be continuing the same form of phenomenographic analysis with a larger and more diverse pool of interviewees. Furthermore, some of these interviewees will be systems programmers and other people who use Unix in their employment. If circumstances permit, we may also interview academics at other institutions. By interviewing a larger and more diverse set of people, we hope to capture a rich picture of peoples' conceptions of Unix, and the reasons why it should be taught.

Beyond unix, this paper demonstrates how phenomenography can be used as a tool for syllabus design in general. It can be used to define various positions, before debating the pros and cons of the positions. Meetings that debate the design of new degrees are highly charged emotionally. Academics who are not inclined to join such a difficult debate can be encouraged to articulate their position as part of an early, non-confrontational, data gathering phenomenographic study. Beginning with a phenomenographic study may therefore lead to a more inclusive and comprehensive approach to syllabus design in general.

References

Åkerlind, G. (2005) Variation and commonality in phenomenographic research methods. *Higher Education Research & Development*. **24**(4): 321-334.

Astrachan, O., Bruce, K., Koffman, E., Kölling, M., Reges, S. (2005) "Resolved: Objects Early Has Failed". SIGCSE'05, February 23-27, 2005, St. Louis, Missouri, USA.

Bach, M. J. (1986) *The Design of the UNIX Operating System*. Englewood Cliffs, NJ: Prentice-Hall. ISBN: 0132017997.

Berglund, A. (2005) *Learning computer systems in a distributed project course: The what, why, how and where*. Acta Universitatis Upsaliensis, Uppsala Dissertations from the Faculty of Science and Technology 62. ISBN 91-554-6187-5.

Biggs, J. B. & Collis, K. F. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. New York, Academic Press, 1982.

Booth, S. (1992). *Learning to program: A phenomenographic perspective.* PhD thesis, University of Gothenberg, Sweden.

Booth, S. (2001) Learning to Program as Entering the Datalogical Culture: a Phenomenographic Exploration. In *9th European Conference for Research on Learning and Instruction (EARLI),* Fribourg, Switzerland.

Booth, S. and Ingerman, A. (2002) Making sense of Physics in the first year of study. *Learning and Instruction* **12:** 493-507

Box, I., & Lister, R. (2005). Variation in students' conceptions of object-oriented information system development. In O. Vasilecas, A. Caplinskas, W. Wojtkowski, W. G. Wojtkowski, J. Zupancic & S. Wrycza (Eds.), *Information systems development: Advances in theory, practice and education. Proceedings of 13th international conference on information systems development (ISD 2004) Vilnius, Lithuania, September 9–11 2004* (pp. 439–451): Springer.

Bruce, C. and Gerber, R. (1995) Towards university lecturers' conceptions of student learning. *Higher Education*, **29**, 443-458

Bruce, C, Buckingham, L, Hynd, J, McMahon, C, Roggenkamp, M, and Stoodley, I. (2004) Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University. *J. of Information Technology Education*, **3**: 143-159. http://jite.org/documents/Vol3/v3p143-160-121.pdf [accessed May 2005]

Bruce, K. (2005) Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list. ACM SIGCSE Bulletin. Volume 37, Issue 2 (June 2005) 111-117.

Cope, C. (2002) Seeking Meaning: The Educationally Critical Aspect of Learning About Information Systems, *Proceedings of the Informing Science + IT Education Conference,Cork,Ireland*.http://proceedings.informingscience.org/IS2002Proceedings/papers/cope190seeki.pdf [accessed Apr. 2006]

Doyle, B. and Lister, R. (2006) A Preliminary Phenomenographic Study Concerning Student Experiences of Unix. *Proceedings of the 19th Annual Conference of the NACCQ*. Wellington NZ 7th-10th July 2006 pp 73-78

A. Eckerdal, A. & Thun, M. (2005) Novice Java Programmers' Conceptions of "Object" and "Class", and Variation Theory. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pp. 89-93.

Kutay, C. and Lister, R. (2006) Up Close and Pedagogical: Computing Academics Talk about Teaching. *Conferences in Research in Practice in Information Technology*, **52**.

Lewis, T., and Smith, W. (2005) The Computer Science Debate: It's a Matter Perspective. SIGCSE Bulletin. Volume 37, Issue 2 (June 2005) 80-84.

Lister, R., Box, I., Morrison, B., Tenenberg, J., Westbrook, S. (2004) The Dimensions of Variation in the Teaching of Data Structures. *9th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, Leeds, UK, 28-30 June. pp.

Lister, R., Berglund, A., Box, I., Cope, C., Pears, A., Avram, C., Bower, M., Carbone, A., Davey, B., de Raadt, M., Doyle, B., Fitzgerald, S., Mannila, L., Kutay, C., Peltomäki, M., Sheard, J., Simon, Sutton, K., Traynor, D., Tutty, J., Anne Venables, A. (2007) Differing Ways that Computing Academics Understand Teaching. Conferences in Research in Practice in Information Technology, **66**.

McCauley, R. (2004) Thinking about our teaching. ACM SIGCSE Bulletin, Vol. 36, Issue 2 (June), 18-19

Reges, S. (2006) Back to Basics in CS1 and CS2. Proceedings of the 37th Technical Symposium on Computer Science Education (SIGCSE 2006). Houston, Texas, USA. pp. 293-297.

Reid, A. (1997), The Meaning of Music and the Understanding of Teaching and Learning in the Instrumental Lesson, in *Proceedings of the Third Triennial ESCOM Conference*, ed. A. Gabrielsson, Uppsala, Sweden: European Society for the Cognitive Sciences of Music, **3**, 200-205.

Reid, A. and Petocz, P. (2003) Completing the Circle: Researchers of Practice in Statistics Education. *Mathematics Education Research J.*, **15**(3): 288-300.

Stoodley, I. Christie, R. and Bruce, C. (2004) Masters Students' Experiences of Learning to Program: An Empirical Model. *Proceedings of QualIT2004: International Conference on Qualitative Research.* http://sky.fit.qut.edu.au/~bruce/pub/papers/QualIT2004-Bruce.pdf [accessed October 2006]

Trigwell, K. (2000) *Phenomenography: Discernment and Variation* http://www.learning.ox.ac.uk/files/Phenom_ISL_paper.pdf [accessed Mar. 2006]

Trigwell, K. and Prosser, M.(1997), Towards an Understanding of Individual Acts of Teaching and Learning (1997) Higher Education Research and Development, 16(2): 241-252.